**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

# КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

**на тему:**

## «Інформаційно-комунікаційна технологія керування інфраструктурою мережі»

**Завідувач випускаючої кафедри** — **Довбиш А. С.**

**Керівник роботи** — **Великодний Д. В.**

**Студента групи ІН.м-02** — **Щербака М. Ю.**

**СУМИ 2021**

*Сумський державний університет*

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«122 - Комп'ютерні науки»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

# ЗАВДАННЯ

# НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

*Щербаку Михайлу Юрійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційно-комунікаційна технологія керування інфраструктурою мережі*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи)_____

3. Вхідні данні до проекту (роботи) _____

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

*1) інформаційний огляд; 2) постановка завдання; 3) визначення методів рішення завдання; 4) проектування роботи; 4) реалізація поставлених задач; 5) аналізи результатів роботи.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *1) Вступ; 2) Інформаційний огляд; 3) Вимоги до функціоналу; 4) Програмна реалізація; 5) Висновки;*

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

# КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів дипломного проекту (роботи) | Термін виконання проекту (роботи) | Примітка |
|---|---|---|---|
| 1. | Огляд методів уніфікації процесів та програмних рішень для керування процесами | | |
| 2. | Постановка задачі та формування завдань дослідження. | | |
| 3. | Проектування інформаційної системи | | |
| 4. | Програмна реалізація інформаційної системи | | |
| 5. | Оформлення пояснювальної записки до кваліфікаційної магістерської роботи | | |

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

# РЕФЕРАТ

**Записка:** 45 стор., 31 рис., 0 табл., 2 додатки, 20 джерел.

**Об’єкт дослідження** — адміністрування мережі на базі Cisco пристроїв.

**Мета роботи** — розробка Telegram боту для адміністрування мережі.

**Методи дослідження** — метод розробки телеграм боту з використанням Telegram Bot API..

**Результати** — розроблений Telegram бот для керування пристроями мережі з Cisco OS.

ТЕЛЕКОМУНІКАЦІЇ, МЕРЕЖА, TELEGRAM, БОТ, PYTHON, TELEBOT, CISCO

# ЗМІСТ

# ВСТУП

Мережа є невід'ємною частиною IT індустрії. Важко уявити сучасну IT компанію, що не використовує мережі для реалізації передачі даних між пристроями. Планування, розробка та налаштування мереж – достатньо глобальні теми, які потребують знань професіоналів різних галузей. Кожна мережа складається з апаратного та програмного забезпечення і не може працювати без будь-якої з частин. Тому, наприклад, інженер телекомунікаційних технологій досі є затребуваною професією. Ефективність кваліфікованого інженера телекомунікаційних технологій залежить не тільки від його знань та навичок. Збільшити продуктивність також може комфорт спеціаліста, на який впливає зручність використовуваних технологій.

Соціальні мережі на данний момент найзручніший та найпростіший спосіб комунікації. А звичний додаток найчастіше сприймається зручним інструментом, адже використовується щодня. Так за даними опитування компанії Research & Branding Group від 10-21 липня 2021 року протягом останніх чотирьох років кількість користувачів Facebook, YouTube, Instagram та Telegram в Україні зросла. Зокрема, найбільші темпи зростання – утричі – у Telegram. Telegram отримав свою популярність не тільки за захищеність даних користувачів, а й за різноманітність функціоналу. Окрім спілкування у чатах з одним співрозмовником або конференціях до 100 000 осіб, Telegram надає можливість вести та переглядати канали, здійснювати дзвінки. Для користувачів месенджеру, які тісно пов'язані з IT галуззю Telegram надає безкоштовне API для роботи з ботами. Боти - це сторонні додатки, які працюють у Telegram. Вони виглядають як акаунти, якими замість людей керують програми. Основне завдання бота - автоматично відповідати після команди, введеної користувачем. Працюючи безпосередньо через

інтерфейс Telegram, програма імітує дії живого користувача, завдяки чому використання такого бота є зручним і зрозумілим.

Таким чином, Telegram надає можливість розробити бота для покращення комфорту вирішення задач адміністрування мережі. Також важливим аспектом є можливість бота надавати доступ до мережі з будь-якого пристрою, що підтримує додаток Telegram.

Отже, метою даної роботи є розробка Telegram боту для адміністрування пристроїв мережі.

# 1. ІНФОРМАЦІЙНИЙ ОГЛЯД

## 1.1 Огляд операційних систем для мережевих пристроїв

Як і звична для нас техніка, така як комп’ютер, смартфон, планшет, мережеві пристрої мають свої операційні системи. Помилково їх сприймають не як повноцінні операційні системи, а лише програму, що керує роботою пристрою. Оперційна система умовного маршрутизатора не дає можливості користуватися звичними для ОС додатками, таким як текстовий або графічний редактори. Але все одно вона залишається операційною систему, що передбачена для ефективного контролю маршрутів пакетів інформації.

Сучасні операційні системи мають дві основні задачі: управління ресурсами та логічне розділення між апаратною та програмною частинами обчислювальної системи. Ця відмінність надає розробникам інтерфейс між прикладними програмами та апаратним забезпеченням комп’ютера. Це означає, що програмісту не потрібно досконально знати апаратне забезпечення: адже програмний код, який безпосередньо спілкується з різними пристроями, вже написаний і вбудований в операційну систему. Іншою функцією операційної системи є управління ресурсами комп’ютера. Наприклад, робочим часом центрального процесору, пам’яттю або дисковим простором. Це централізоване управління дозволяє кільком додаткам ефективно використовувати одні й ті ж обчислювальні ресурси. Як і у випадку з апаратним забезпеченням, програміст звільняється від необхідності вбудовувати будь-який програмний код, який керує ресурсами комп’ютера, у свої програми.

Раніше самі маршрутизатори вважалися суто апаратними пристроями. Поділу між програмним і апаратним забезпеченням майже не було.

Зі зростанням популярності мереж виникла потреба в маршрутизаторах, які підтримують різні протоколи та забезпечують додаткову

функціональність, наприклад комутацію. В результаті сучасні програмні рішення мережевих пристроїв перетворилися на багатофункціональні операційні системи, які підтримують маршрутизацію та комутацію.

Чим відрізняється операційна система мережевого пристрою? Її функції повністю спрямовані на виконання певного класу завдань. Операційна система повинна забезпечувати найпростіші завдання – IP-маршрутизацію та її адміністрування, підтримку IP-телефонії, доступ до OSPF (Open Shortest Path First) протоколу. Для роботи ОС достатньо досить простого комп’ютера. Мінімальні вимоги: не менше 64 МБ пам’яті, частота шини материнської плати не менше 100 МГц, жорсткий диск об’ємом 64 Мб і більше. Багатопроцесорні системи не підтримуються, так само, як не підтримуються системи SCSI, USB, RAID. Жорсткий диск з встановленою системою можна перенести на інший комп'ютер, але копіювати і переносити встановлену систему не можна - так працює система ліцензування.

### 1.1.1 Junos OS

Junos OS (Juniper Network) - це операційна система на основі FreeBSD, яка використовується в апаратних маршрутизаторах Juniper Networks. Juniper пропонує пакет розробки програмного забезпечення (SDK) для партнерів та клієнтів, щоб забезпечити додаткове налаштування [1].

Junos OS розроблена з урахуванням автоматизації, програмування та DevOps. Операційна система має API бібліотеку та інтеграцію з популярними фреймворками автоматизації. Junos підтримує REST API, NETCONF і YANG. Інфраструктура потокової телеметрії передає дані пристрою, системи та мережі в режимі реального часу за допомогою кількох моделей експорту даних, таких як OpenConfig і gRPC.

Підтримка автоматизації Junos OS:

- Потокова телеметрія та інтерфейс телеметрії Junos (JTI). JTI забезпечує доступ до даних у режимі реального часу за допомогою потокової телеметрії. Використовуючи підхід моделі push, вона є більш ефективною, розширюваною та масштабованою, ніж модель SNMP. Підтримка JTI галузевих стандартів, таких як OpenConfig і gRPC, спрощує інтеграцію з операційним середовищем.
- Відкриті API. Інтерфейс Junos OS REST API дозволяє безпечно підключатися до пристроїв Junos, виконувати віддалені виклики процедур і використовувати графічний інтерфейс REST API Explorer. Окрім NETCONF і YANG, Junos підтримує протокол Path Computation Element (PCEP) для встановлення, керування та видалення конфігурацій мережевих пристроїв.
- Junos Extension Toolkit (JET). JET надає сучасний програмний інтерфейс для розробки додатків і спрощення автоматизації операційних, конфігураційних та завдань управління.

### 1.1.2 Cisco IOS

Cisco IOS (Internetwork Operating System) — це операційна система, яка працює на маршрутизаторах та комутаторах Cisco Systems. Основною функцією Cisco IOS є забезпечення передачі даних між вузлами мережі. Окрім маршрутизації та комутації, Cisco IOS пропонує додаткові послуги, які адміністратор може використовувати для підвищення продуктивності та безпеки мережевого трафіку. Ці послуги включають шифрування, аутентифікацію, глибоке тестування пакетів, якість обслуговування (QoS), інтелектуальну маршрутизацію та можливості проксі. На маршрутизаторах Cisco Integrated Services (ISR) iOS також може підтримувати обробку викликів та послуги уніфікованих комунікацій [2].

Основні елементи системи:

- Процеси. Зазвичай під процесами розуміються окремі потоки і пов'язані з ними дані. Процеси виконують конкретні завдання, такі як підтримання працездатності системи, комутацію мережних пакетів та реалізацію протоколів маршрутизації.
- Ядро системи здійснює основні функції операційної системи: управління пам'яттю та планування завдань, а також відповідає за розподіл апаратних ресурсів (пам'ять та центральний процесор) між усіма процесами.
- Буфери пакетів. Зазвичай це буфери пам'яті, що використовуються для зберігання мережевих пакетів, що маршрутизуються. Драйвери пристроїв.
- Драйвери керують апаратною частиною мережевих інтерфейсів та периферійними пристроями (такими як флеш-карти). Драйвер виступає у ролі посередника між ядром системи IOS з усіма процесами та апаратною частиною маршрутизатора. Драйвери також безпосередньо взаємодіють із програмним забезпеченням швидкого перемикання пакетів. Програмне забезпечення швидкого перемикання пакетів. Під таким програмним забезпеченням мається на увазі набір оптимізованих функцій, що здійснюють швидке перемикання шляхів проходження пакетів.

Cisco вважається провідною компанією в світі у галузі телекомунікацій, а їх пристрої та програмне забезпечення найбільш поширені порівнючи з конкурентами. Через це наразі існує велика кількість якісних і зручних бібліотек, реалізованих для роботи з Cisco IOS.

### 1.2 Огляд реалізацій існуючих рішень

Серед існуючих рішень можна виділити Telegram бот “Pekko” (Рисунок 1.1). Даний бот націлений на автоматизацію роботи мережевого адміністратора, використовуючи функціонал бота.

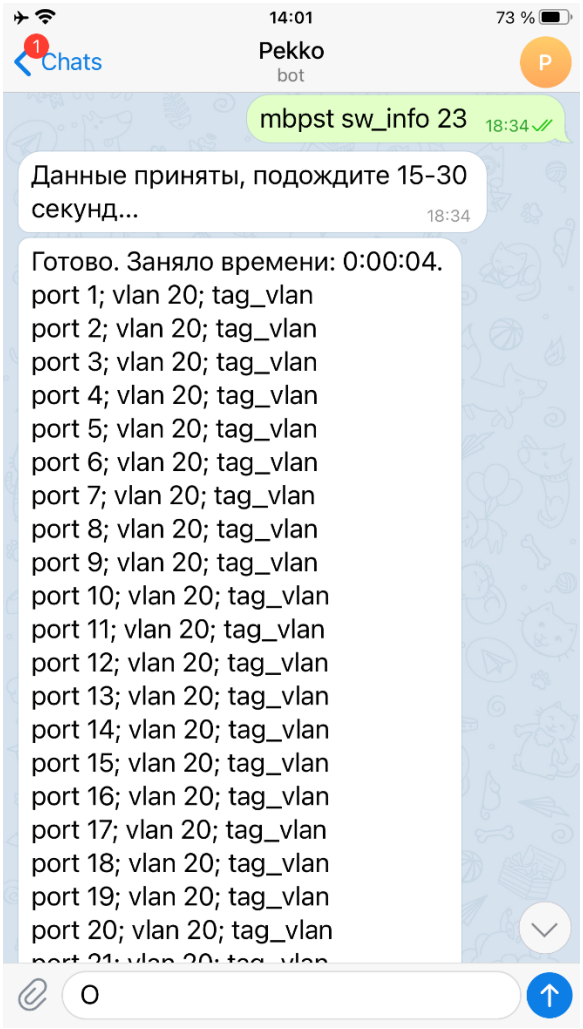


Рисунок 1.1 – Приклад використання Telegram бота “Pekko”

З переваг Telegram бота “Pekko” можна виділити можливість аудиту портів комутатора доступу та збереження конфігурацій (Рисунок 1.2).

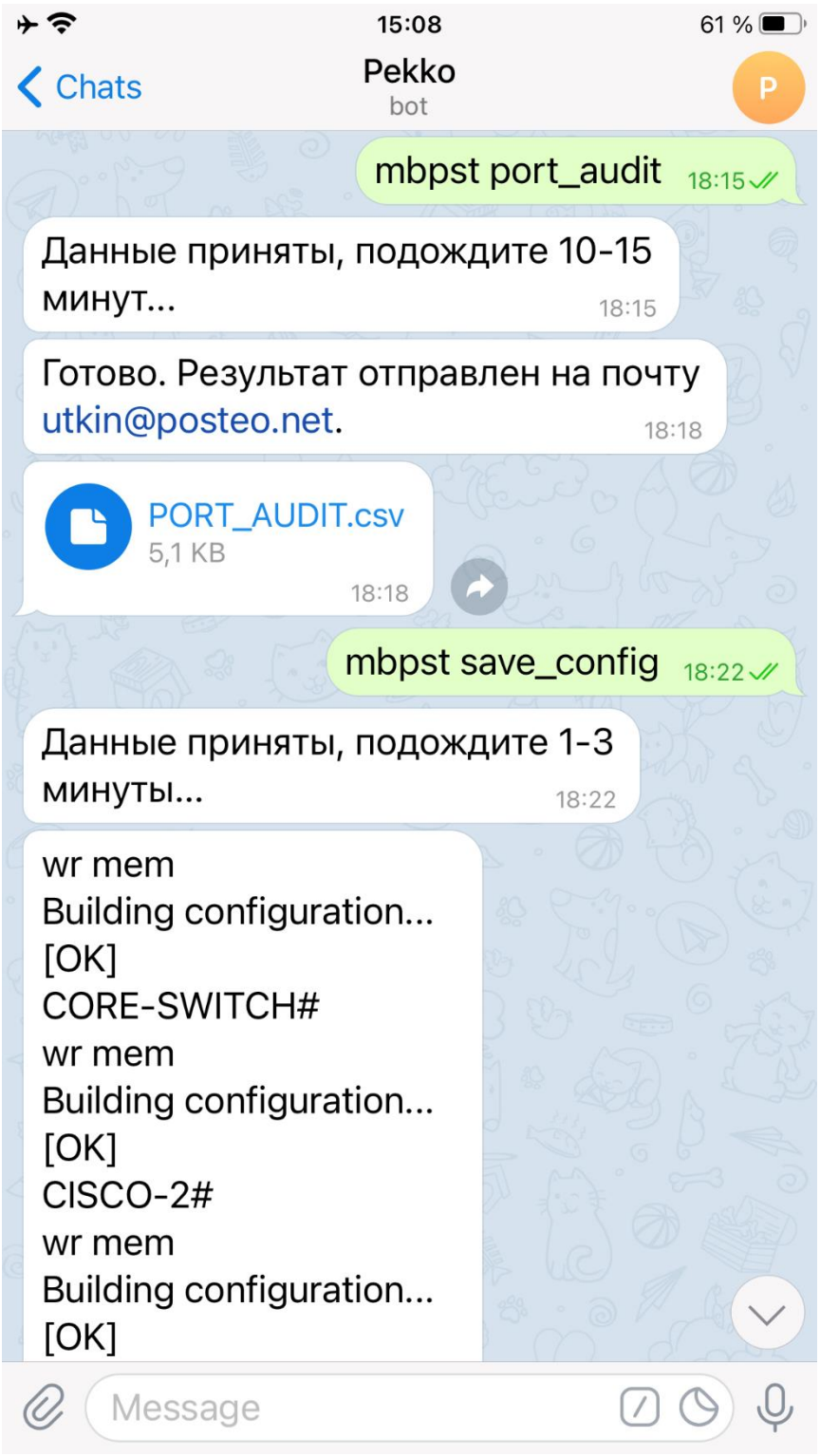


Рисунок 1.2 – Приклад функцій аудиту та зберігання конфігурацій

### 1.3 Постановка задачі

Метою даної роботи є розробка Telegram боту для керування мережі на базі пристроїв Cisco. Бот повинен надавати можливість виконувати наступні операції:

- Додавати та виконувати підключення до Cisco пристроїв у мережі
- Відображати поточні підключення
- Виконувати команди для налаштування або моніторингу Cisco пристроїв у мережі
- Обробляти та повертати результати виконання команд для налаштування та моніторингу Cisco пристроїв у мережі
- Видаляти поточні сесії окремих Cisco пристроїв у мережі

Описати кроки відтворення кожного сценарію та зробити висновок про виконану роботу.

# 2. ТЕХНОЛОГІЇ РОЗВ'ЯЗКУ ПОСТАВЛЕНОЇ ЗАДАЧІ

## 2.1 GNS3

Реальні пристрої, що використовуються для побудування мережі, мають досить велику ціну та не завжди є доступними у повсякденності. Враховуючи, що компанія Cisco є провідною у виготовленні мережевого обладнання та підтримки їх програмного забезпечення і відрізняється гарною якість та стабільністю, ціна на її продукти може бути ще вищою в залежності від певного пристрою. Не маючи можливості отримати доступ до реальних пристроїв було вирішено використовувати програмний емулятор GNS3. Головними перевагами цього емулятора є безкоштовність та майже ідентичний функціонал програмного забезпечення мережевих приладів.

GNS3 дозволяє запускати не тільки образи Dynamips, а й QEMU, а також взаємодіяти з IOU та іншими віртуальними машинами. Додаток запускається безпосередньо на машині, де знаходиться. Для емуляції пристроїв воно може використовувати віртуальні машини, розташовані на цьому ж хості або віддалено. Підтримується робота в Linux, Windows та Mac OS X. Великим плюсом GNS3 є можливість використання тих самих інструментів, що й для роботи з реальними пристроями: PuTTY, SecureCRT, Wireshark та ін.

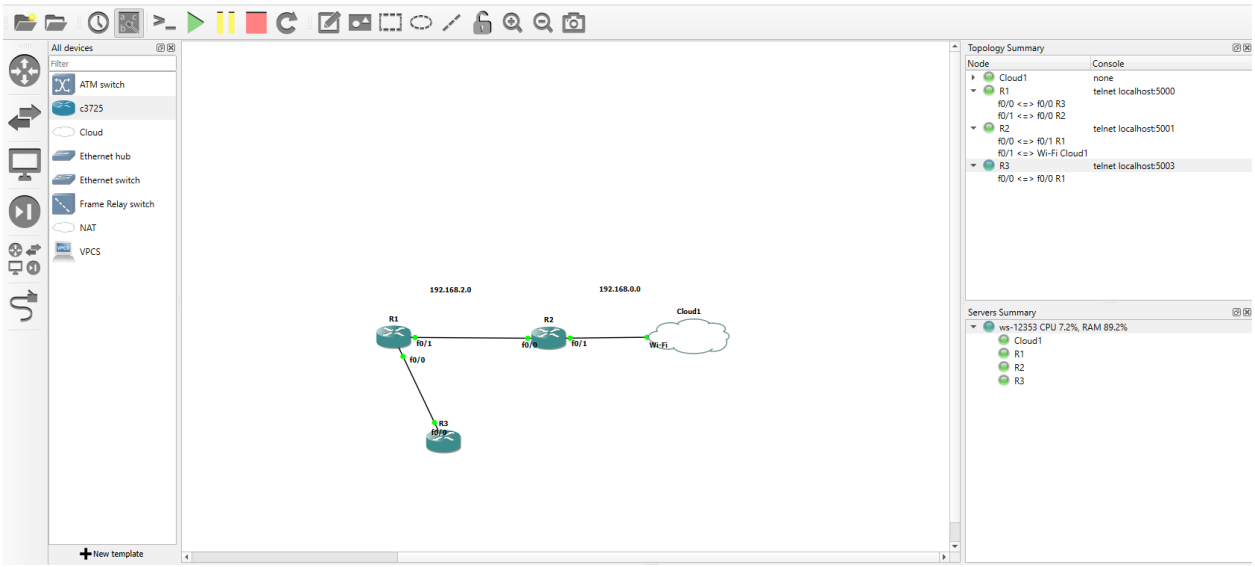


Рисунок 2.1 – Графічний інтерфейс GNS3

## 2.2 PuTTY

PuTTY — клієнтська програма для роботи із мережевими протоколами Telnet, SSH, SCP, SFTP, для підключення по COM-порту та ZModem, утиліта для генерації RSA, DSA, ECDSA, Ed25519 цифрових SSH-ключів. PuTTY є вільним додатком з відкритим вихідним кодом, що містить реалізацію мережевих протоколів SSH, Telnet, Rlogin та розповсюджується під ліцензією Open Source MIT.

Пакет PuTTY включає в себе кілька додатків:

- PuTTY: Telnet і SSH клієнт
- PSCP: SCP клієнт — копіювання файлів за шифрованим протоколом з управлінням з командної строки
- PSFTP: SFTP клієнт — копіювання файлів по SSH, як FTP
- PuTTYtel: окремий клієнт Telnet
- Plink: інтерфейс командного рядку до PuTTY
- Pageant: агент SSH-аутентифікації для PuTTY, PSCP і Plink
- PuTTYgen: утиліта для генерації SSH-ключей [3]

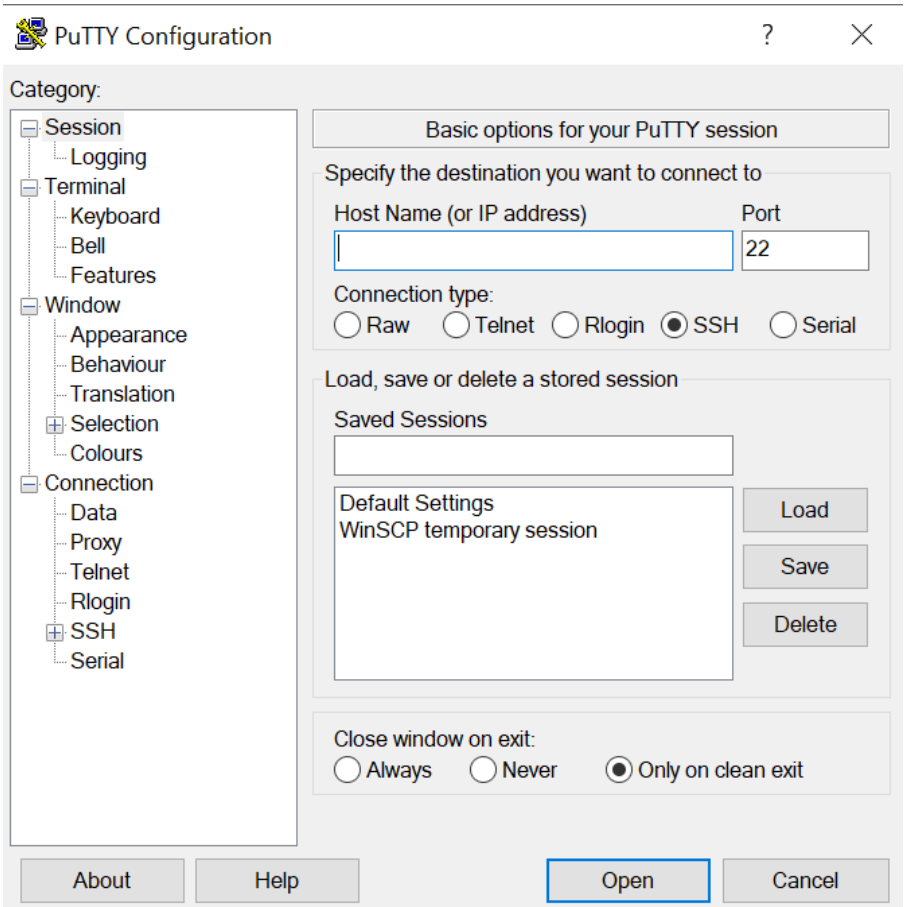


Рисунок 2.2 – Графічний інтерфейс PuTTY

### 2.3 SSH

SSH (Secure Shell) - мережевий протокол, що використовується для віддаленого керування операційними системами та передачі файлів. Його особливість у тому, що SSH шифрує трафік, роблячи підключення безпечними. За замовчуванням використовує 22 порт. Програмні реалізації SSH поділяються на серверні та клієнтські частини. Найчастіше, як сервер застосовується OpenSSH, як клієнт - OpenSSH (UNIX), PuTTY або SecureCRT (Windows, Linux). Для підключення необхідна IP-адреса пристрою та обліковий запис. Для отримання повного списку ключів можна скористатись командою man ssh. Незалежно від операційної системи сімейства UNIX (Ubuntu, CentOS, FreeBSD тощо) для налаштування сервера SSH

використовується конфігураційний файл /etc/ssh/sshd_config. Для Windows також існує реалізація сервера - OpenSSH for Windows, яка дозволить підключатися до даної системи для віддаленого управління з командного рядка.

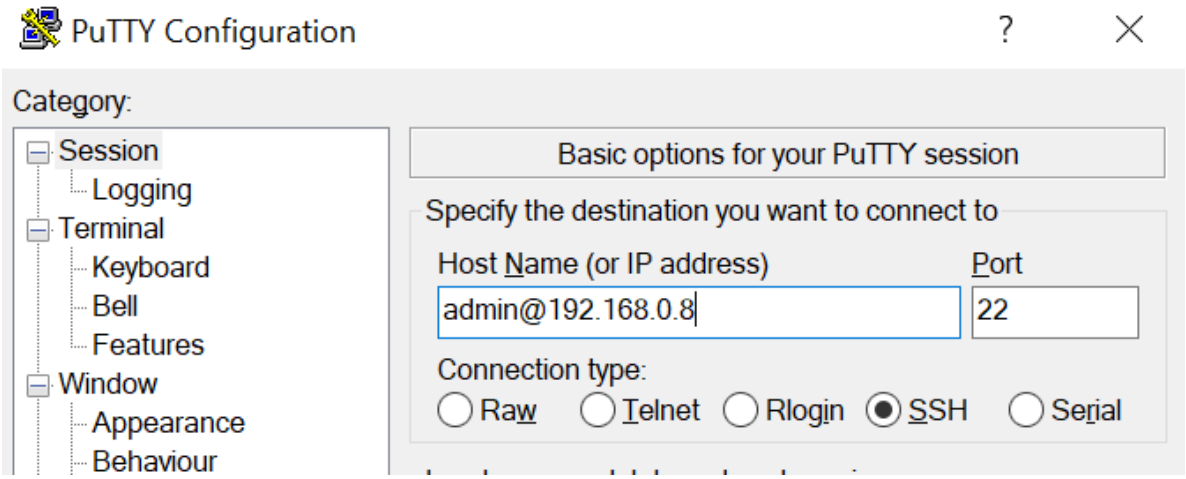


Рисунок 2.3 – Приклад з'єднання з сервером використувуючи SSH протокол

Ключ SSH — це облікові дані безпечного доступу, які використовуються в протоколі Secure Shell (SSH). Ключі SSH використовують пари ключів на основі технології інфраструктури відкритих ключів (PKI), золотого стандарту аутентифікації та шифрування цифрової особистості, щоб забезпечити безпечний метод аутентифікації. Оскільки протокол SSH широко використовується для зв'язку в Cloud сервісах, мережевих середовищах, інструментах передачі файлів, інструментах керування конфігурацією та інших залежних від комп'ютера сервісах, більшість організацій використовують ключі SSH для автентифікації особи та захисту цих служб від ненавмисного використання або зловмисних атак. Ключі SSH не тільки покращують безпеку, але й дозволяють автоматизувати підключені процеси, єдиний вхід (SSO), а також керування ідентифікацією та доступом у таких масштабах, яких потребує сучасний бізнес.

Використовуючи криптографічні алгоритми та методи автентифікації PKI, ключі SSH не піддаються атакам зловмисників, а дійсні облікові дані не відкриваються, якщо система або сервер були зламані. Криптографія PKI безперервно вдосконалюється, захищаючи ідентичність у майбутньому від нових та загроз, що розвиваються. А автентифікація ключа SSH зручніша, ніж автентифікація паролем. Ключі SSH з’єднують користувачів і процеси з сервером шляхом автоматичного ініціювання автентифікації та надання доступу, тому користувачам не потрібно запам’ятовувати або вводити свій пароль для кожної системи.

### 2.4 Маршрутизатор c3725

Мережа, емульована для демонстрації роботи телеграм боту складається з трьох маршрутизаторів c3725. Нижче наведені базові характеристики даного пристрою:

- Високопродуктивний 240-МГц процесор комп'ютера зі скороченим набором інструкцій (RISC).
- До 256 МБ SDRAM
- Пам'ять CompactFlash до 128 МБ
- Два слоти для мережевих модулів, один з яких може вмістити мережевий модуль подвійної ширини
- Три слоти для інтерфейсних карт
- Два слоти Cisco 3700 CompactFlash (один зовнішній і один внутрішній)
- Два слоти AIM
- Підтримка системи резервного живлення Cisco

# 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1. Вибір засобів програмної реалізації

### 3.1.1. Мова програмування

Для реалізації Telegram боту “NettyBot” була обрана мова програмування Python. Відмінною рисою мови Python є легкість читання, ясність та висока якість. Код написаний мовою Python читається легше. З цього випливає, що обслуговування написаного коду виконується набагато простіше, ніж програмний код, написаний на інших мовах. Одноманітність оформлення програмного коду на мові Python полегшує його розуміння навіть для тих, хто не брав участі в його створенні. Крім того, Python підтримує сучасні механізми багаторазового використання програмного коду, яким є об'єктно-орієнтоване програмування (ООП).

- Окрім зазначених переваг Python - одна з найкращих мов для досягнення високої швидкості розробки програмного забезпечення. У порівнянні з компілюючими або строго типізованими мовами, такими як C, C ++ і Java, Python у багато разів підвищує продуктивність праці розробника. Обсяг програмного коду на мові Python зазвичай становить 20-30% від еквівалентного коду на мові C ++ або Java. Це означає менший обсяг введення з клавіатури, менша кількість часу на налагодження і менший обсяг трудовитрат на підтримку. Крім того, програми на мові Python запускаються відразу ж, минаючи тривалі етапи компіляції.
- Важливим фактором вибору цієї мови програмування є великий спектр корисних OpenSource пакетів, які досить зручно використовувати для реалізації проектів. Одним з таких пакетів є саме telebot – реалізація Telegram Bot API мовою Python.

- Під час реалізації боту наступні методи з Telegram Bot API були використані:

  1) sendMessage(chat_id, text) – метод для відправлення повідомлення ботом користувачу

  2) messageHandler() – обробник, необхідний для вилову повідомлень від користувачів. [4]

### 3.1.2 Середовище розробки

Для розробки було використано популярне та високо оцінюване інтегроване середовище розробки – PyCharm. Це зручне IDE надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів та багато інших корисних дрібниць, таких як підказка по сигнатурі функції (Рисунок 3.1), знаходження всіх згадувань конструкції (Рисунок 3.2) та інші.[4]

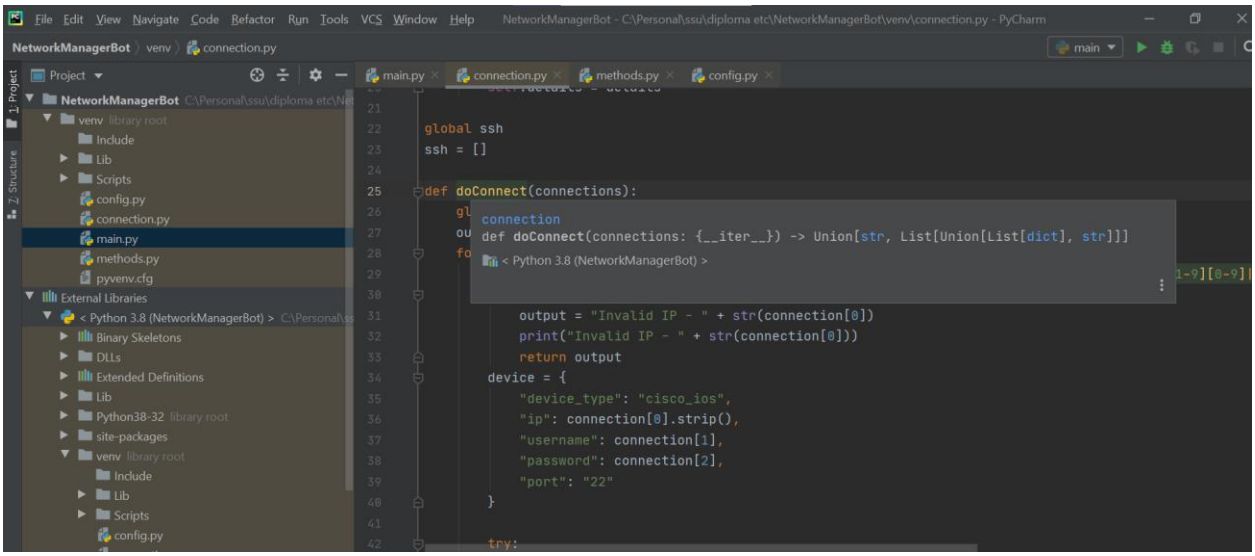


Рисунок 3.1 – Підказка по сигнатурі функції

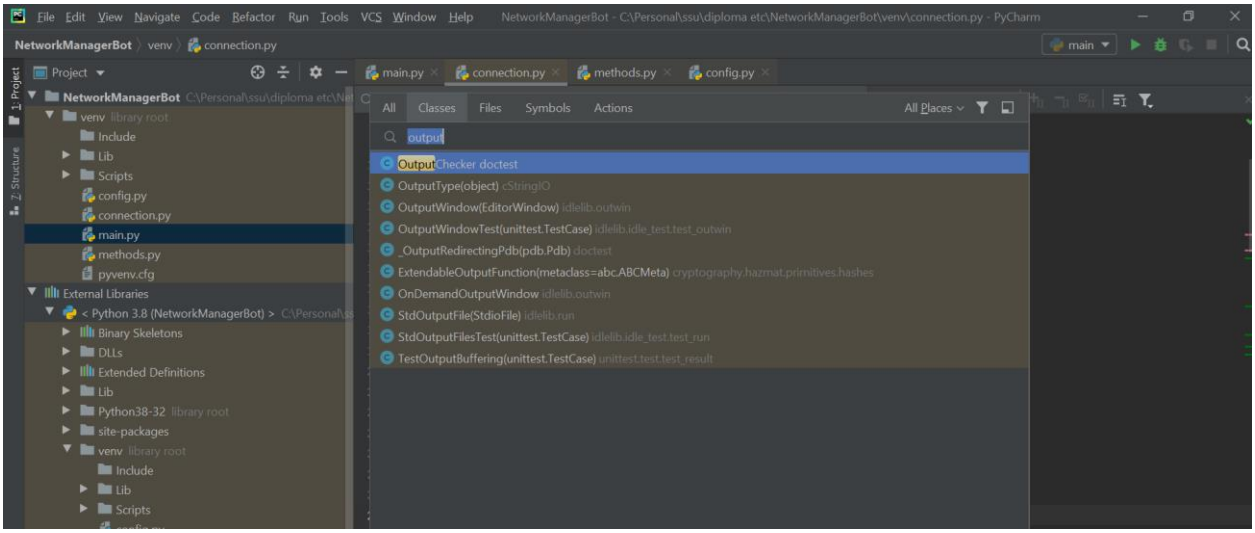


Рисунок 3.2 – Знаходження всіх згадувань конструкції

### 3.2. Опис розроблених методів

Список методів, реалізованих для для роботи телеграм боту:

1) def addConnection(message) – метод приймає повідомлення користувача, що містить інформацію про мережеві пристрої для підключення. Викликає метод def doConnect(connections)
2) def doConnect(connections) – метод приймає массив об’єктів, що містять інформацію про пристрої та виконує підключення через SSH протокол
3) def getConnections() – метод повертає всі активні підключення та відображає їх користувачу у чаті
4) def closeConnection(message) – метод приймає повідомлення користувача, що містить інформацію про мережевий пристрій та виконує відключення
5) def sendCommand(command) – метод приймає повідомлення користувача, що містить назву мережевого пристрою та команду. Відправляє на обраний пристрій необхідну команду

6) def findSsh(device_name) – метод приймає назву мережевого пристрою, задану користувачем при створенні підключення. Виконує пошук ip адреси мережевого пристрою за заданою назвою. Викликається у методі def sendCommand(command).

### 3.3 Розробка мережі в GNS3 для тестування Telegram бота

Згідно з постановкою задачі Telegram бот повинен підтримувати з'єднання з декількома мережевими пристроями одночасно. Тому створимо мережу з трьома маршрутизаторами c3725 та Cloud (Рисунок 3.3 – Мережа з трьома маршрутизаторами c3725 та Cloud) для підключення емульованої мережі до реальної локальної мережі і виконаємо всі необхідні налаштування.

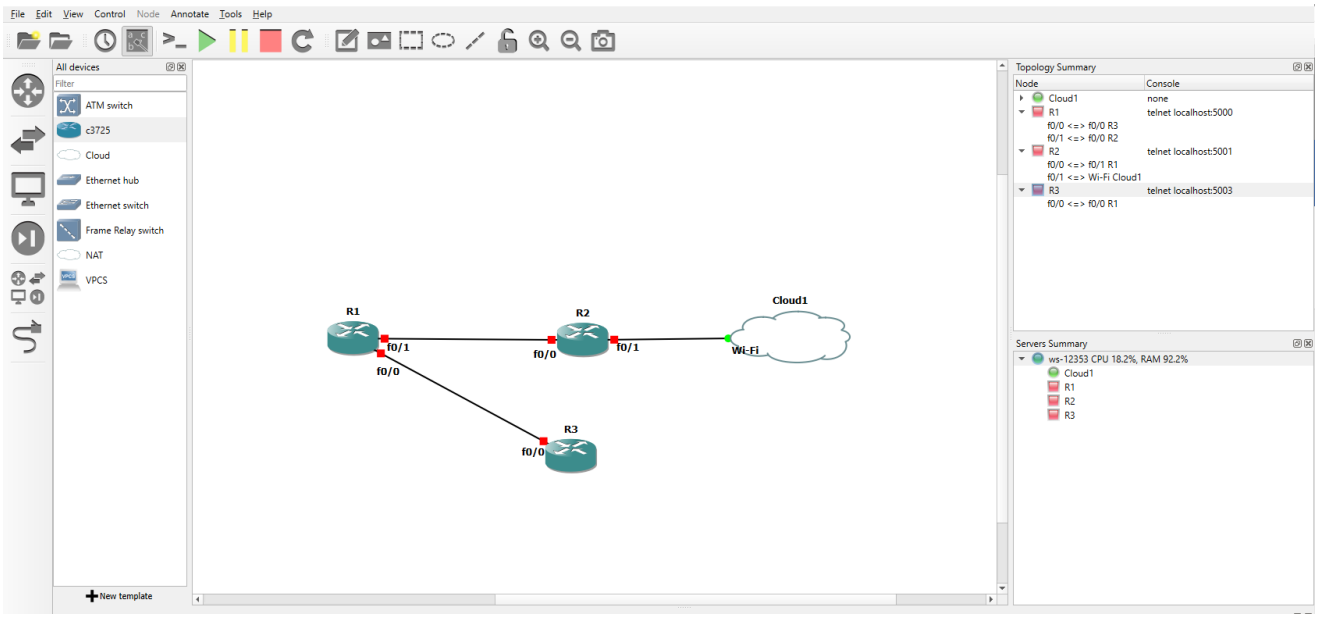


Рисунок 3.3 – Мережа з трьома маршрутизаторами c3725 та Cloud

Налаштування машрутизатора R1:

1) Запуск маршрутизатора кнопкою Start (Рисунок 3.4)

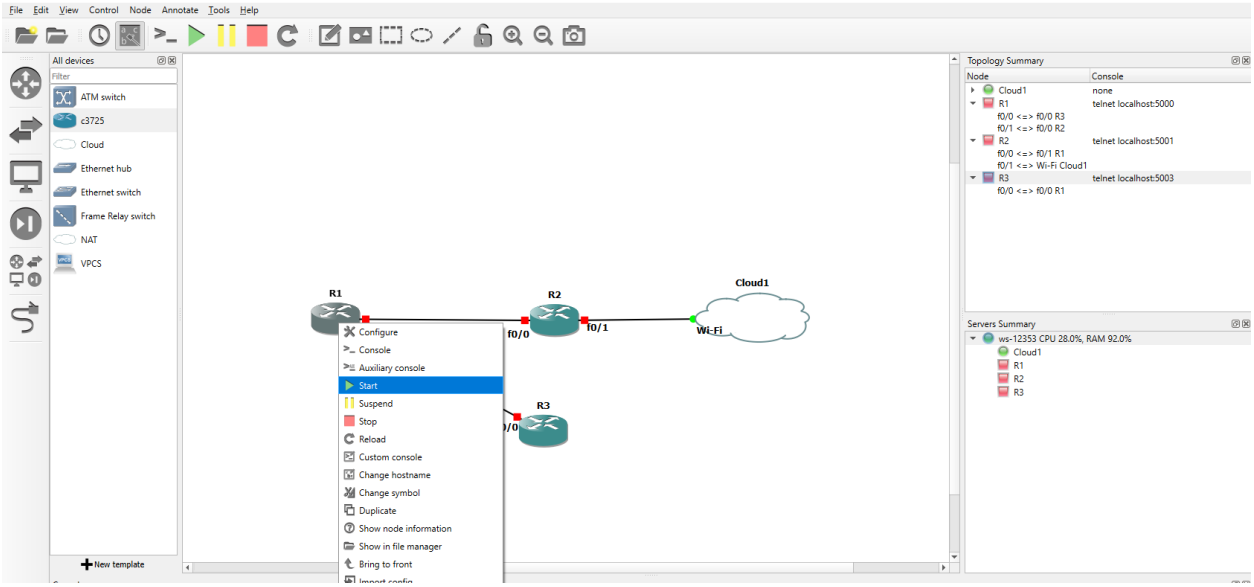


Рисунок 3.4 – Запуск маршрутизатора R1

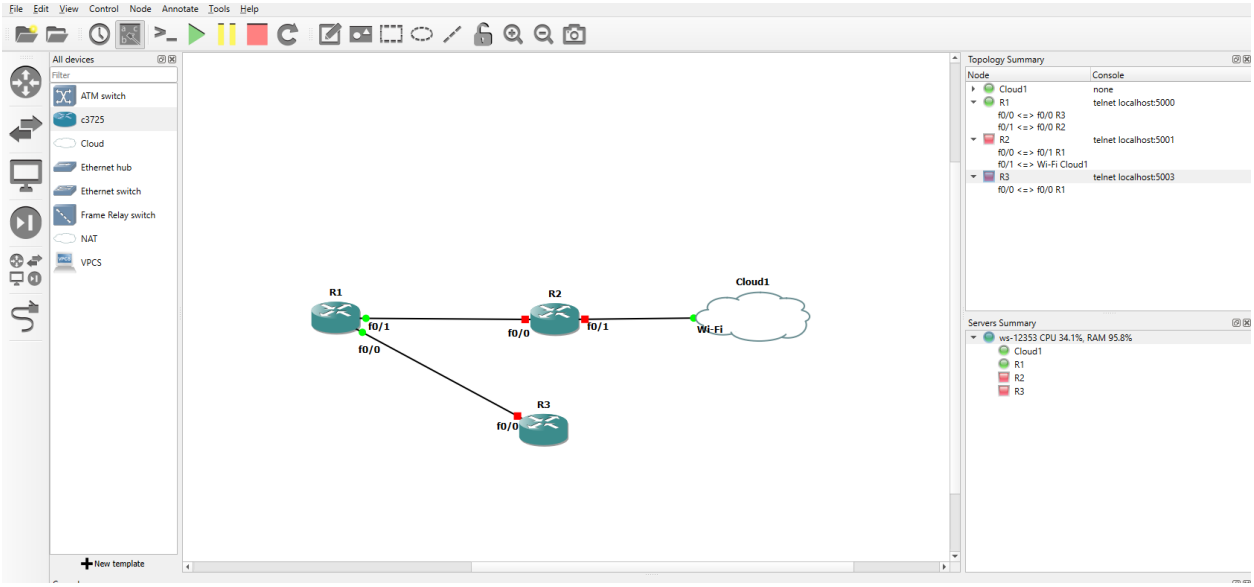


Рисунок 3.5 – Маршрутизатор R1 увімкнено

2) Запуск консолі для налаштування маршрутизатора R1 (Рисунок 3.6)

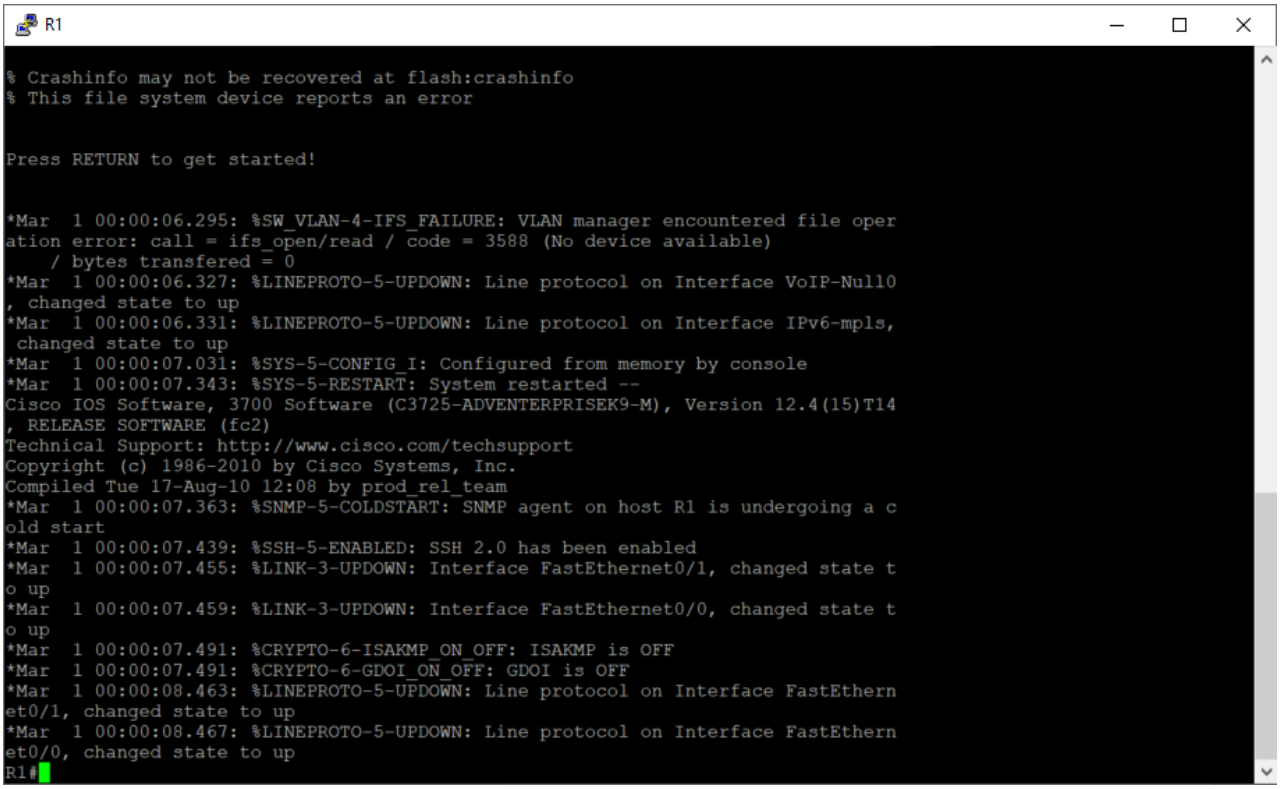


Рисунок 3.6 – Консоль вводу команд для маршрутизатора R1

3) Задаємо ip адрес для двох інтерфейсів маршрутизатора R1, вмикаємо їх та зберігаємо налаштування

R1#conf t

R1(config-if)#int f0/0

R1(config-if)#ip address 192.168.3.1 255.255.255.0

R1(config-if)#no shutdown

R1(config-if)#ine f0/1

R1(config-if)#ip address 192.168.2.1 255.255.255.0

R1(config-if)#no shutdown

R1(config-if)#exit

R1(config)#exit

R1#wr

4) Перевірямо налаштування інтерфейсів маршрутизатора R1 командою ‘sh ip int br’

```
R1#sh ip int br
Interface                  IP-Address      OK? Method Status                Prot
ocol
FastEthernet0/0            192.168.3.1     YES manual up                    up

FastEthernet0/1            192.168.2.1     YES NVRAM  up                    up

R1#
```

Рисунок 3.7 – IP адреси інтерфейсів маршрутизатора R1

5) Задаємо Gateway для маршрутизатора R1 як IP адреса інтерфейсу f0/1 машрутизатора R2

   R1#conf t

   R1(config)#ip route 192.168.0.0 255.255.255.0 192.168.2.2

Повторюємо ті ж самі дії для машрутизаторів R2 та R3, використовуючи різні IP адреси.

```
R2#sh ip int br
Interface                  IP-Address      OK? Method Status                Prot
ocol
FastEthernet0/0            192.168.2.2     YES NVRAM  up                    up

FastEthernet0/1            192.168.0.2     YES NVRAM  up                    up

R2#
```

Рисунок 3.8 – IP адреси інтерфейсів маршрутизатора R2

```
R3#sh ip int br
Interface                  IP-Address      OK? Method Status                Prot
ocol
FastEthernet0/0            192.168.3.2     YES NVRAM  up                    up

FastEthernet0/1            unassigned      YES NVRAM  administratively down down

FastEthernet1/0            unassigned      YES unset  up                    down

FastEthernet1/1            unassigned      YES unset  up                    down

FastEthernet1/2            unassigned      YES unset  up                    down

FastEthernet1/3            unassigned      YES unset  up                    down

FastEthernet1/4            unassigned      YES unset  up                    down

FastEthernet1/5            unassigned      YES unset  up                    down

FastEthernet1/6            unassigned      YES unset  up                    down

FastEthernet1/7            unassigned      YES unset  up                    down

FastEthernet1/8            unassigned      YES unset  up                    down
```

Рисунок 3.9 – IP адреси інтерфейсів маршрутизатора R3

Вводимо наступні команди у командному рядку cmd, щоб налаштувати Gateway для реального Wifi-маршрутизатора у локальній мережі:

route add 192.168.2.0 mask 255.255.255.0 192.168.0.2

route add 192.168.3.0 mask 255.255.255.0 192.168.0.2

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19044.1348]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>route add 192.168.2.0 mask 255.255.255.0 192.168.0.2
 OK!

C:\WINDOWS\system32>route add 192.168.3.0 mask 255.255.255.0 192.168.0.2
 OK!

C:\WINDOWS\system32>
```

Рисунок 3.10 – Налаштування Gateway для реального Wifi-маршрутизатора у локальній мережі

Для успішного підключення до машрутизаторів за допомогою Telegram боту необхідно налаштувати SSH на кожному з пристроїв у мережі. Для цього необхідно:

1) Задати доменне ім’я для ip адреси пристрою
2) Згенерувати ключ авторизації
3) Задати логін та пароль для підключення
4) Задати необхідні привілегії

Усі команди для налаштування SSH знаходяться у Додатку А.

### 3.4 Тестування роботи Telegram боту для керування пристроями мережі

Нижче наведені скріншоти основного функціоналу Telegram бота:

1) Початок роботи з ботом:

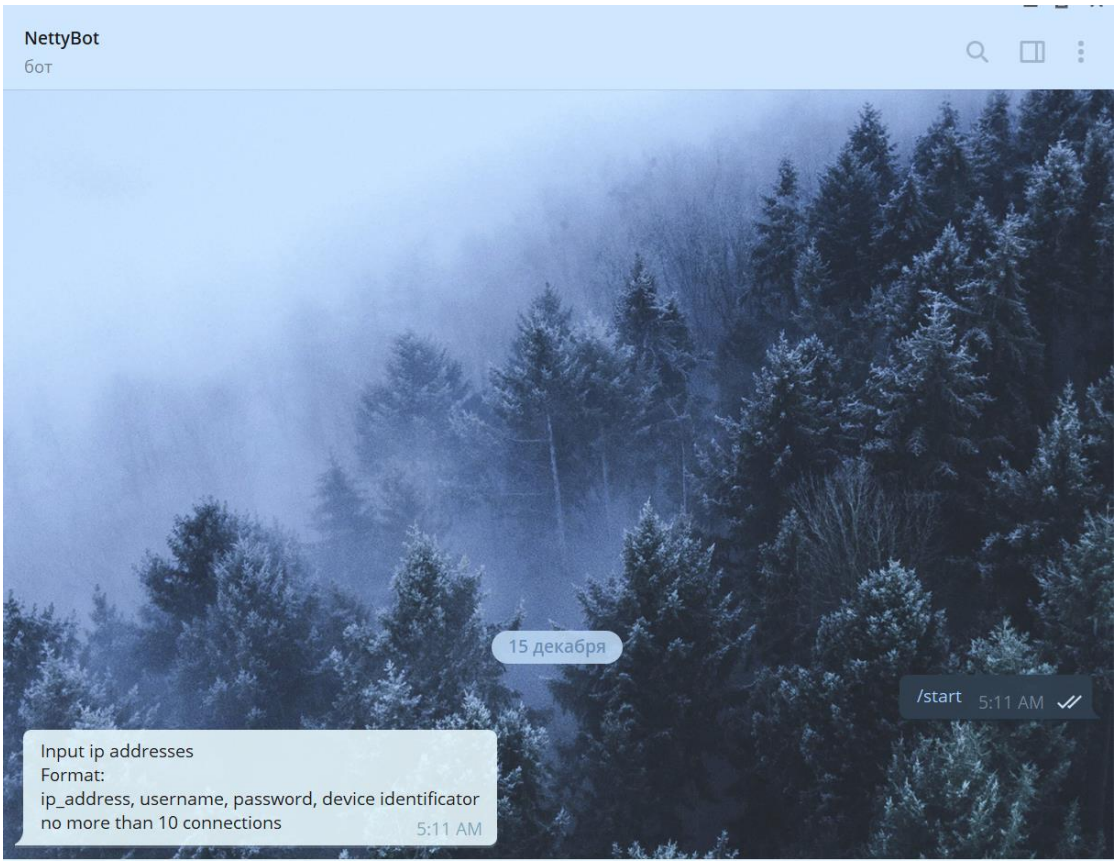


Рисунок 3.11 – Початок роботи з ботом

2) Перевіримо активні підключення командою /get

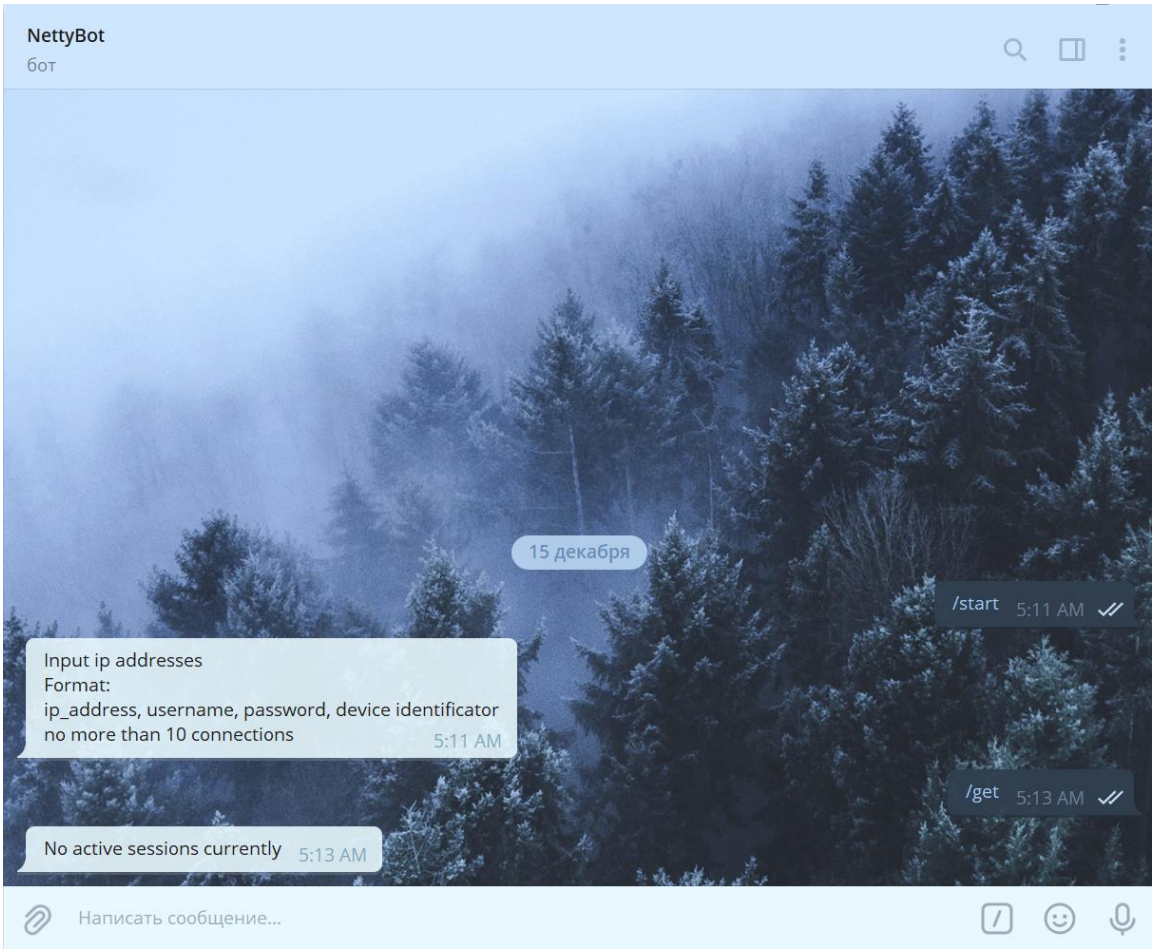


Рисунок 3.12 – Повідомлення про відсутніть активних підключень

3) Добавимо 3 підключення до маршрутизаторів. При успішному підключенні для кожного з пристроїв буде виконана команда “sh ip int br”

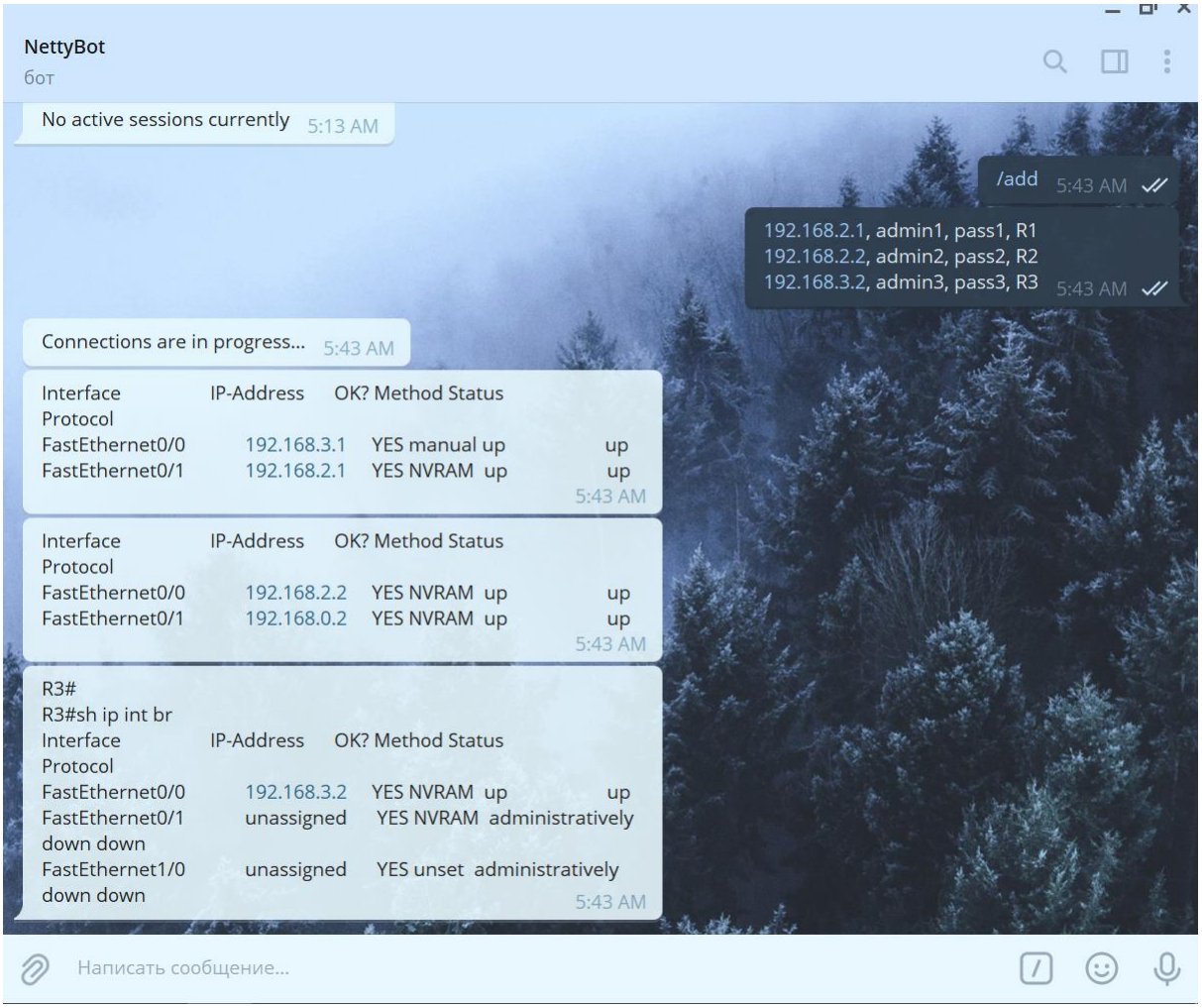


Рисунок 3.13 – Успішне підключення до маршрутизаторів

4) Налаштуємо інтерфейс FastEtherne1/0 для маршрутизатора R3. Для цього необхідно написати назву мережевого пристрою та необхідну команду через відступ.

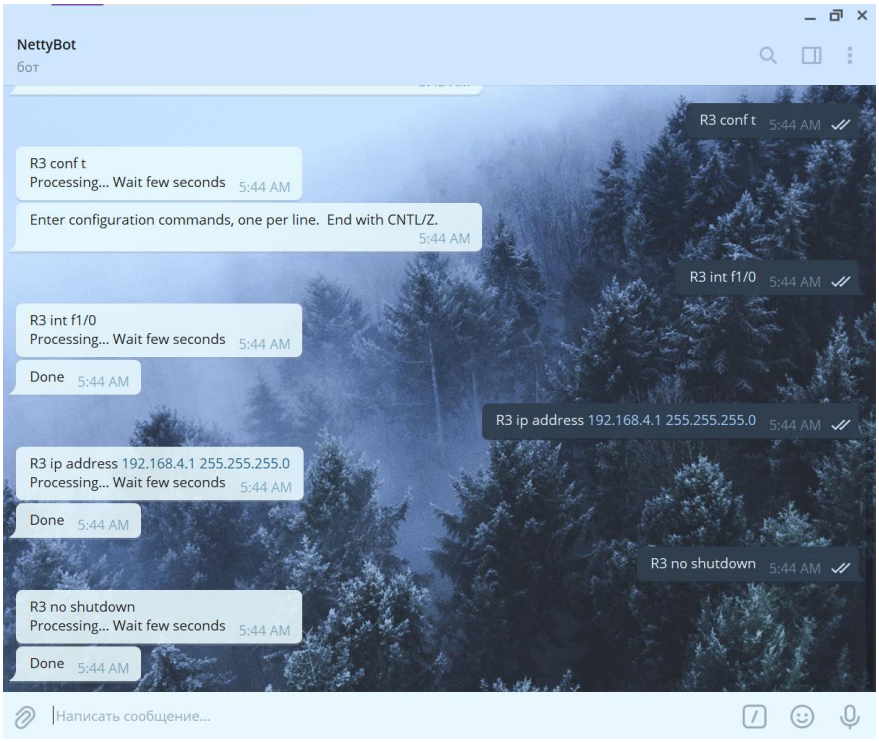


Рисунок 3.14 – Налаштування інтерфейсу 1/0 маршрутизатора R3

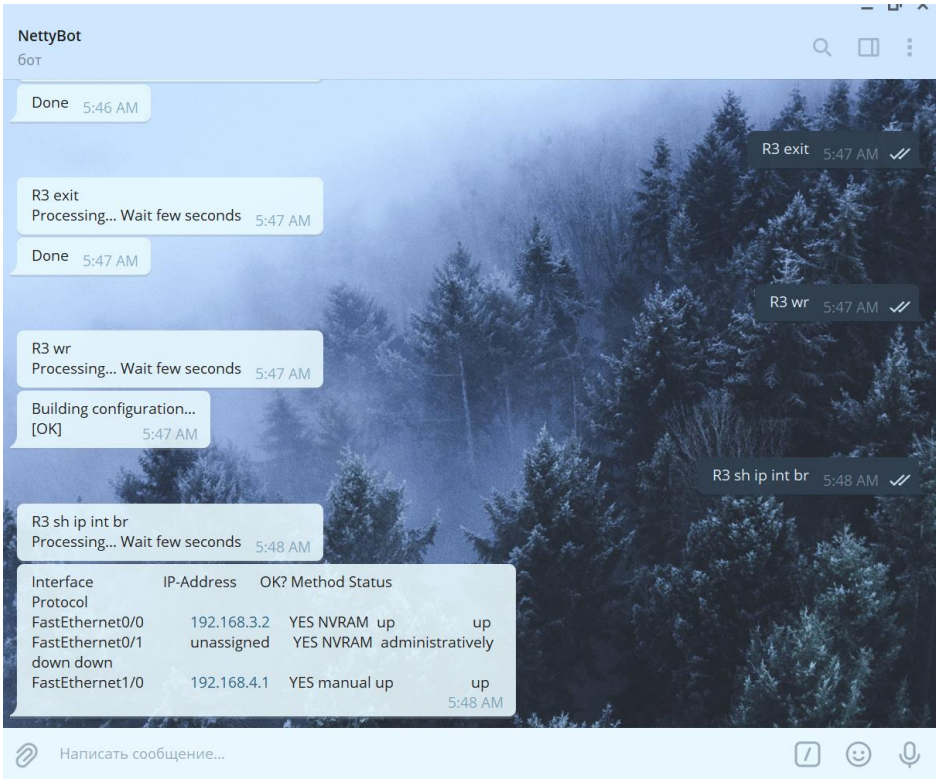


Рисунок 3.15 – Інтерфейс 1/0 маршрутизатора R3 успішно налаштований

5) Перевіримо деякі команди для інших маршрутизаторів

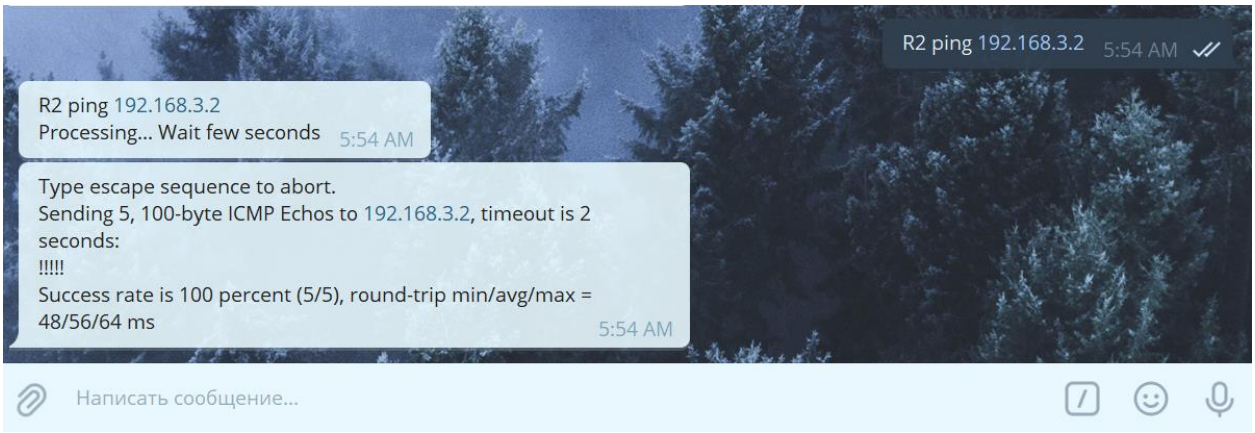


Рисунок 3.16 – Результат виконання ping команди з маршрутизатора R2

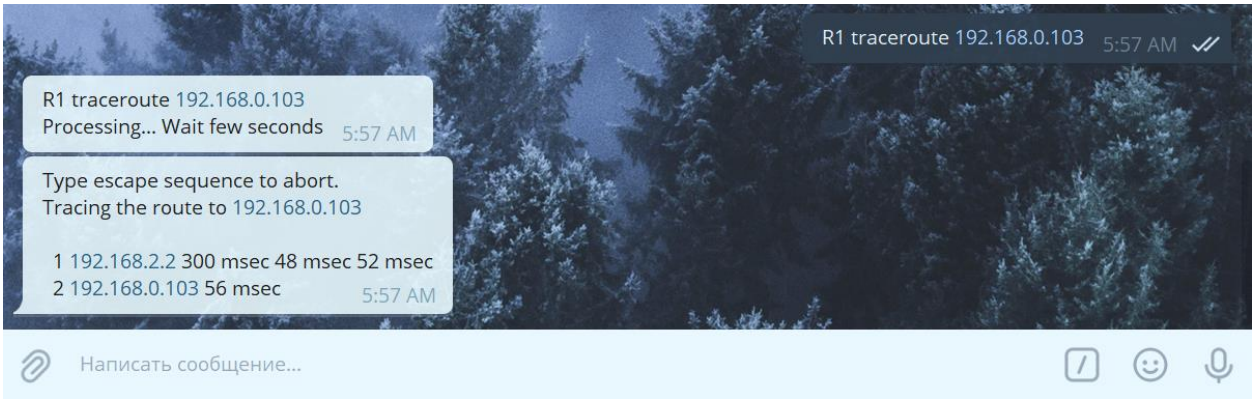


Рисунок 3.17 – Результат виконання traceroute команди з маршрутизатора R1

6) Перевіримо відображення активних підключень за допомогою /get

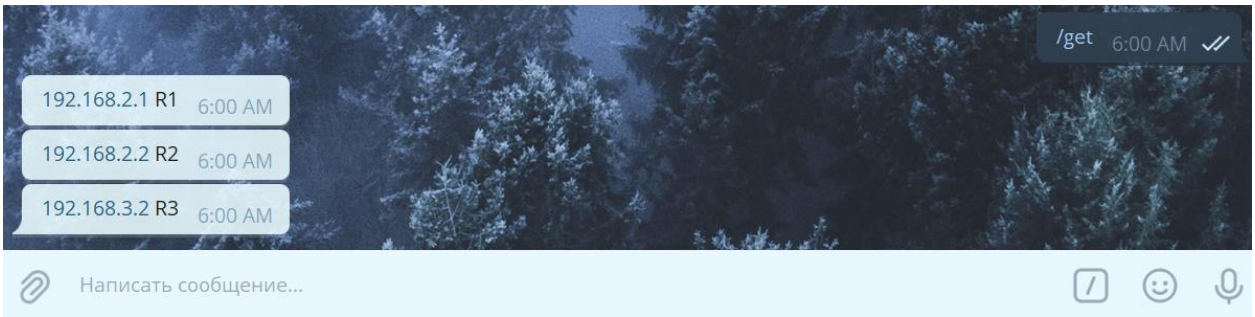


Рисунок 3.18 – Відображення активних підключень

7) Перевіримо відключення з’єднання з маршрутизатором R3

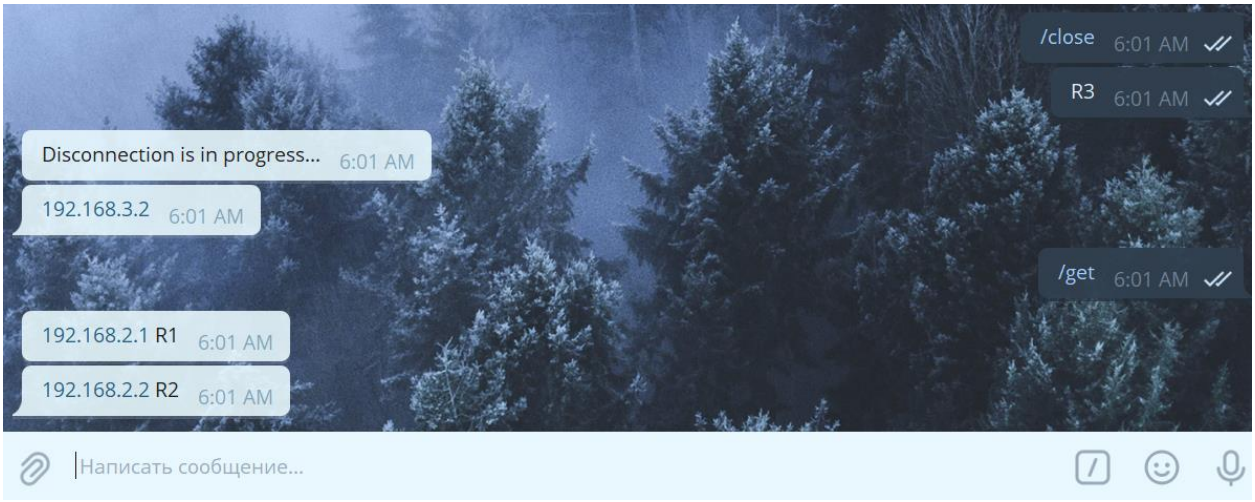


Рисунок 3.19 – З’єднання з маршрутизатором R3 успішно зупинено

8) Перевіримо обробку виключень

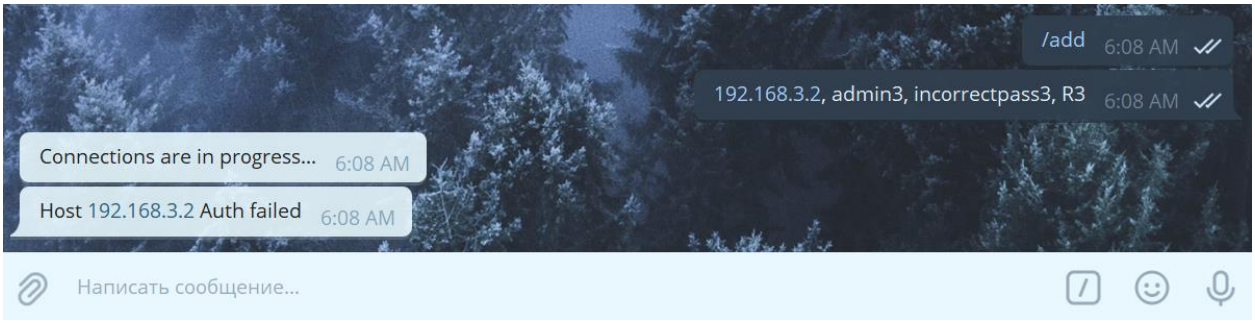


Рисунок 3.20 – “Auth failed” виключення

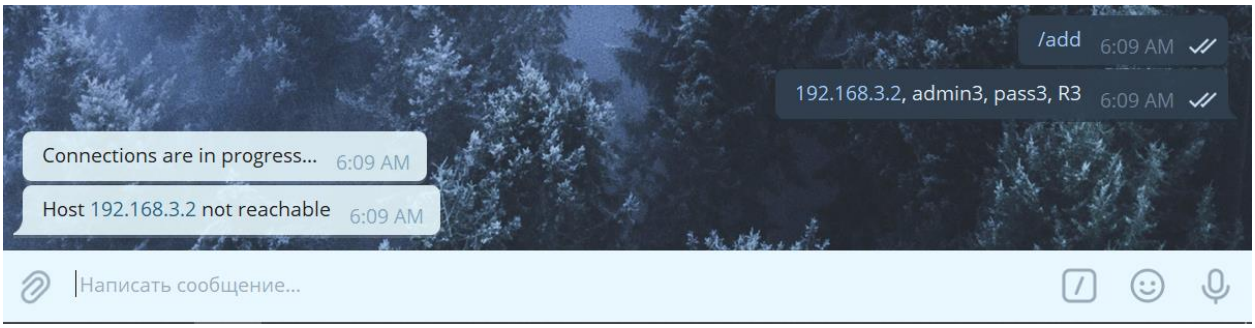


Рисунок 3.21 – “Host not reachable” виключення

# ВИСНОВОК

У результаті виконаної кваліфікаційної магістерської роботи був реалізований Telegram бот для керування пристроями мережі з Cisco OS. Вимоги з постановки задачі виконані. Telegram бот “Netty” підтримує можливість додавати, видаляти та переглядати підключення мережевих пристроїв за протоколом SSH. Бот може виконувати CLI команди для будь якого пристрою, з яким він з’єднаний. Комунікація з ботом відбувається за допомогою чату. Результати виконання команд відправляються користувачеві у приватному повідомленні.

За час виконання роботи поглибив знання у програмуванні мовою Python, а також проектуванні та адмініструванні мереж. Поглибив навички користування емулятором GNS3 та програмою PuTTY.

З можливих майбутніх покращень можна виділити:

- збільшення спектру підтримуваних пристроїв з програмним забезпеченням на базі різних операційних систем. Наприклад, Junos OS, MikroTik RouterOS та інші;
- автоматизацію деяких часто використовуваних команд за допомогою графічного інтерфейсу Telegram;
- моніторинг мережі для виявлення проблем та їх категорізацію.

# СПИСОК ЛІТЕРАТУРИ


1. Juniper Networks Junos OS [Електронний ресурс] — Режим доступу: https://www.ibm.com/docs/en/dsm?topic=networks-juniper-junos-os (дата звернення 17.11.2021) – Назва з екрана
2. Cisco IOS Technologies [Електронний ресурс] — Режим доступу: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-technologies/index.html (дата звернення 18.11.2021) – Назва з екрана
3. PuTTY: Telnet/SSH Клиент [Електронний ресурс] — Режим доступу: https://putty.org.ru (дата звернення 17.11.2021) – Назва з екрана
4. Щербак Михайло. Телеграм бот для аналізу прогнозу погодних умов. 2020
5. Олексій Васильєв. Програмування мовою Python. 2019
6. Real Python Course, Part 1 Real Python Team. Real Python. 2017
7. Telegram Bot API [Електронний ресурс] — Режим доступу: https://core.telegram.org/bots/api (дата звернення 15.11.2021) – Назва з екрана.
8. Python Pros and Cons: What are The Benefits and Downsides of the Programming Language [Електронний ресурс] — Режим доступу: https://www.netguru.com/blog/python-pros-and-cons (дата звернення 15.11.2021) – Назва з екрана.
9. Benefits of Python over Other Programming Languages [Електронний ресурс] — Режим доступу: https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/ (дата звернення 15.11.2021) – Назва з екрана.
10. Python 3.8.3 documentation [Електронний ресурс] — Режим доступу: https://docs.python.org/3/ (дата звернення 16.11.2021) – Назва з екрана.

11. What is PyCharm? Features, Advantages & Disadvantages [Електронний ресурс] — Режим доступу: https://hackr.io/blog/what-is-pycharm (дата звернення 16.11.2021) – Назва з екрана.
12. PyCharm Features [Електронний ресурс] — Режим доступу: https://www.jetbrains.com/pycharm/features/ (дата звернення 17.11.2021) – Назва з екрана.
13. Соцмережі як джерело інформації українців [Електронний ресурс] — Режим доступу: https://rb.com.ua/blog/socseti-kak-istochnik-informacii-ukraincev/ (дата звернення 17.11.2021) – Назва з екрана.
14. What Is an SSH Key? [Електронний ресурс] — Режим доступу: https://sectigo.com/resource-library/what-is-an-ssh-key (дата звернення 19.11.2021) – Назва з екрана.
15. Todd Lammle. CCNA: Cisco Certified Network Associate: Review Guide 978-1-118-06346-0
16. ISDN Overview [Електронний ресурс] — Режим доступу: http://hea-www.harvard.edu/~fine/ISDN/overview.html (дата звернення 16.11.2021) – Назва з екрана.
17. Управління мережевою інфраструктурою [Електронний ресурс] — Режим доступу: https://habr.com/ru/post/568114/ (дата звернення 19.11.2021) – Назва з екрана.
18. GNS3 Academy [Електронний ресурс] — Режим доступу: https://gns3.teachable.com/courses (дата звернення 16.11.2021) – Назва з екрана
19. Overview of Cisco 3700 Series [Електронний ресурс] — Routershttps://www.andovercg.com/datasheets/cisco-3700-routers.pdf (дата звернення 16.11.2021) – Назва з екрана

20. Cisco IOS Release 12.2(15)ZJ [Електронний ресурс] — Режим доступу: https://www.cisco.com/c/en/us/td/docs/ios/12_2/12_2z/release/notes/rn3700zj.html (дата звернення 16.11.2021) – Назва з екрана

# ДОДАТОК А. КОМАНДИ ДЛЯ НАЛАШТУВАННЯ МЕРЕЖЕВИХ ПРИСТРОЇВ

```
R1(config-if)#int f0/0
R1(config-if)#ip address 192.168.3.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#in f0/1
R1(config-if)#ip address 192.168.2.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#ip route 192.168.0.0 255.255.255.0 192.168.2.2
R1(config)#exit
R1#wr

R2(config-if)#int f0/0
R2(config-if)#ip address 192.168.2.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#in f0/1
R2(config-if)#ip address 192.168.0.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#ip route 192.168.3.0 255.255.255.0 192.168.2.1
R2(config)#exit
R2#wr

R3(config-if)#int f0/0
R3(config-if)#ip address 192.168.3.2 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#ip route 192.168.0.0 255.255.255.0 192.168.3.1
R3(config)#ip route 192.168.0.0 255.255.255.0 192.168.3.1
R3(config)#exit
R3#wr

R1(config)#ip domain-name dimploma1.com
R1(config)#crypto key generate rsa modulus 1024
R1(config)#ip ssh version 2
R1(config)#ip ssh time-out 15
R1(config)#ip ssh authentication-retries 2
R1(config)#username admin1 privilege 15 password pass1
R1(config)#line vty 0 4
R1(config-line)#login local
R1(config-line)#transport input ssh
R1(config-line)#privilege level 15
R1(config-line)#exit
R1(config)#exit
R1#wr
```

```
R2(config)#ip domain-name dimploma2.com
R2(config)#crypto key generate rsa modulus 1024
R2(config)#ip ssh version 2
R2(config)#ip ssh time-out 15
R2(config)#ip ssh authentication-retries 2
R2(config)#username admin2 privilege 15 password pass2
R2(config)#line vty 0 4
R2(config-line)#login local
R2(config-line)#transport input ssh
R2(config-line)#privilege level 15
R2(config-line)#exit
R2(config)#exit
R2#wr

R3(config)#ip domain-name dimploma3.com
R3(config)#crypto key generate rsa modulus 1024
R3(config)#ip ssh version 2
R3(config)#ip ssh time-out 15
R3(config)#ip ssh authentication-retries 2
R3(config)#username admin3 privilege 15 password pass3
R3(config)#line vty 0 4
R3(config-line)#login local
R3(config-line)#transport input ssh
R3(config-line)#privilege level 15
R3(config-line)#exit
R3(config)#exit
R3#wr
```

# ДОДАТОК Б. ПРОГРАМНИЙ КОД

main.py

```
import config
import telebot
from telebot import types
import connection
import methods

bot = telebot.TeleBot(config.token)

global newConnExpected
global closeConnExpected

@bot.message_handler(commands=['start', 'help'])
def handle_start(message):
    if message.from_user.id == 354303518:
        setConnExpected(True)
        setCloseConnExpected(False)
        info_message = "Input ip addresses\n" \
                       "Format:\n" \
                       "ip_address, username,
password, device identificator\n" \
                       "no more than 10
connections"
        bot.send_message(message.chat.id,
info_message)
    else: bot.send_message(message.from_user.id,
"0JLQuNC50LTQuCDQvtGC0YHRjtC00LAsINGA0L7Qt9Cx0ZbQudC90L
jQug==")

def log(message, answer):
    from datetime import datetime
    print(datetime.now())
    print("Message from {0} {1}. (id = {2}) \n Text
- {3}".format(message.from_user.first_name,

message.from_user.last_name,

str(message.from_user.id),
```

```
message.text))
        print(answer)

    @bot.message_handler(commands=['add'])
    def handle_start(message):
        log(message, "")
        if message.from_user.id == 354303518:
            setConnExpected(True)
            setCloseConnExpected(False)
        else: bot.send_message(message.from_user.id,
"0JLQuNC50LTQuCDQvtGC0YHRjtC00LAsINGA0L7Qt9Cx0ZbQudC90L
jQug==")

    @bot.message_handler(commands=['close'])
    def handle_start(message):
        log(message, "")
        if message.from_user.id == 354303518:
            setConnExpected(False)
            setCloseConnExpected(True)
        else: bot.send_message(message.from_user.id,
"0JLQuNC50LTQuCDQvtGC0YHRjtC00LAsINGA0L7Qt9Cx0ZbQudC90L
jQug==")

    @bot.message_handler(commands=['get'])
    def handle_start(message):
        log(message, "")
        if message.from_user.id == 354303518:
            answer = connection.getConnections()
            if answer == "No active sessions
currently":
                bot.send_message(message.from_user.id,
answer)
            else:
                for ans in answer:

bot.send_message(message.from_user.id, ans)
                print(answer)
```

```
        else: bot.send_message(message.from_user.id,
"0JLQuNC50LTQuCDQvtGC0YHRjtC00LAsINGA0L7Qt9Cx0ZbQudC90L
jQug==")

    @bot.message_handler(content_types=['text'])
    def handle_start(message):
        if message.from_user.id == 354303518:
            close_flag = closeConnExpected
            if close_flag:
                bot.send_message(message.from_user.id,
"Disconnection is in progress...")
                answer =
connection.closeConnection(message)
                setCloseConnExpected(False)
                bot.send_message(message.from_user.id,
answer)
                log(message, answer)
                return
            flag = newConnExpected
            answer = ""
            if flag:
                bot.send_message(message.from_user.id,
"Connections are in progress...")
                answer =
connection.addConnection(message)
                for ans in answer:

bot.send_message(message.from_user.id, ans)
                setConnExpected(False)
                #bot.send_message(message.from_user.id,
message)
                log(message, answer)
            else:
                bot.send_message(message.from_user.id,
message.text + "\nProcessing... Wait few seconds")
                answer = methods.sendCommand(message)
                bot.send_message(message.from_user.id,
answer)
                log(message, answer)
```

```
        else: bot.send_message(message.from_user.id,
"0JLQuNC50LTQuCDQvtGC0YHRjtC00LAsINGA0L7Qt9Cx0ZbQudC90L
jQug==")



    def setConnExpected(a):
        global newConnExpected
        newConnExpected = a

    def setCloseConnExpected(a):
        global closeConnExpected
        closeConnExpected = a

    bot.polling(none_stop=True, interval=0)
```

connection.py

```
    from netmiko import ConnectHandler
    from netmiko import ssh_exception
    from getpass import getpass
    from netmiko.ssh_exception import
NetMikoTimeoutException
    from netmiko.ssh_exception import
NetMikoAuthenticationException
    from netmiko import Netmiko
    import multiprocessing
    import re



    class Ssh:
        ip = ""
        connect = ""
        details = ""

        def __init__(self, ip, connect, details):
            self.ip = ip
            self.connect = connect
            self.details = details
```

```
    global ssh
    ssh = []

    def doConnect(connections):
        global ssh
        output = []
        for connection in connections:
            ip_check = re.findall("^(([0-9]|[1-9][0-
9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-
9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$", connection[0])
            if ip_check == []:
                output = "Invalid IP - " +
str(connection[0])
                print("Invalid IP - " +
str(connection[0]))
                return output
            device = {
                "device_type": "cisco_ios",
                "ip": connection[0].strip(),
                "username": connection[1],
                "password": connection[2],
                "port": "22"
            }

            try:
                net_connect = ConnectHandler(**device)
                output_text =
net_connect.send_command_timing("sh ip int br")
                output.append(output_text)
                net_connect.enable()
                new_ssh = Ssh(connection[0].strip(),
net_connect, connection[3])
                if ssh is None:
                    ssh[0] = new_ssh
                else: ssh.append(new_ssh)
            except
ssh_exception.AuthenticationException:
                output.append("Host " +
connection[0].strip() + " Auth failed")
```

```
                print("Host " + connection[0].strip() +
" Auth failed")
            except
ssh_exception.NetmikoTimeoutException:
                output.append("Host " +
connection[0].strip() + " not reachable")
                print("Host " + connection[0].strip() +
" not reachable")
        return output

    def addConnection(message):
        output = ""
        connections = []
        conn_lines = message.text.splitlines()
        global ssh
        duplicate_checker = ssh
        for line in conn_lines:
            if line == "": continue
            if line[0] == "#": continue
            conn_data = line.split(',')
            ip_addr = conn_data[0].strip()
            username = conn_data[1].strip()
            password = conn_data[2].strip()
            device_ind = conn_data[3].strip()
            for duplicate in duplicate_checker:
                if ip_addr == duplicate.ip:
                    output = "Such connection already
active"
                    print(output)
                    return output
                if device_ind == duplicate.details:
                    output = "Device with such
identificatior already connected. Use a different
device name"
                    print(output)
                    return output
            connections.append((ip_addr, username,
password, device_ind))
        output = doConnect(connections)
        return output
```

```
    def closeConnection(message):
        output = ""
        global ssh
        for connection in ssh:
            if connection.details == message.text:
                output = connection.ip
                connection.connect.disconnect()
                ssh.remove(connection)
                return output
        output = "No connection with such device to be
closed"
        return output

    def getConnections():
        global ssh
        output = []
        none_check = ssh
        print(none_check)
        if none_check == []:
            answer = "No active sessions currently"
            return answer
        for connection in ssh:
            output.append(connection.ip + " " +
connection.details)
        return output

    def getSsh():
        global ssh
        response = ssh
        return response
```

methods.py

```
    import connection

    def sendCommand(command):
        com_data = command.text.split(' ')
        device_name = com_data[0].strip()
        conn = findSsh(device_name)
```

```
        com_to_execute =
command.text.replace(device_name+' ', '')
        if conn == "No device with such name":
            return conn
        else:
            output =
conn.connect.send_command_timing(com_to_execute)
            if output == "":
                output = "Done"
        return output

    def findSsh(device_name):
        con = connection.getSsh()
        result = ""
        for ssh in con:
            if ssh.details == device_name:
                result = ssh
                return result
        if result == "":
            result = "No device with such name"
            return result
```